Eurographics Conference on Visualization (EuroVis) (2012)
M. Meyer and T. Weinkauf (Editors)

Continuous Navigation of Nested Abstraction Levels

Matthew van der Zwan,[1] Alexandru C. Telea,[1] and Tobias Isenberg[1,2]

[1]Johann Bernoulli Institute of Mathematics and Computer Science, University of Groningen, The Netherlands
[2]DIGITEO in collaboration with VENISE–LIMSI–CNRS and AVIZ–INRIA, Saclay, France

Abstract

We investigate the dedicated control of multiple levels of semantic and sampling-based abstraction in 3D datasets, i. e., different types of data abstractions as opposed to sampling-based abstraction which shows more or less data. This dedicated navigation in the abstraction space facilitates the mental integration of different existing visualization techniques in many application areas including our example domain of fluid simulation. We realize the continuous abstraction control by interpolating between the levels while being able to simultaneously show multiple abstractions. We employ a halo-like shading technique based on distance fields to blend between several levels while continuously navigating between focus and context abstractions. We further add a semantic lens to find focus abstractions close to a user-defined context abstraction. Our entire implementation uses 2D image-based techniques to enable real-time performance, which seamlessly integrates within a 3D visualization tool.

Categories and Subject Descriptors (according to ACM CCS): I.3.m [Computer Graphics]: Miscellaneous—Scientific visualization; volumetric flow visualization; illustrative visualization; dedicated abstraction control.

1. Introduction

Abstraction is a—if not even *the*—fundamental principle employed in virtually all areas of visualization because it allows us to uncover and understand principles about the subject matter that we visualize, rather than just seeing the raw data. As Rautek et al. [RBGV08] note, abstraction can be introduced in a visualization either implicitly by selecting a certain style of depiction ("low-level visual abstractions") or explicitly by employing means such as focus+context or distortion ("high-level visual abstractions"). The latter group of high-level abstractions are of particular interest because they are created to emphasize specific chosen aspects of interest to the viewers. Often, however, there exist many different means to achieve explicit or high-level abstractions, all of which are valid and show different important aspects of the same dataset. Therefore it is essential that we can link these different types of abstraction with each other [Duk04] to allow viewers to understand the relationship between them.

In this paper we focus on addressing this issue of combining multiple different layers of abstraction of the same dataset specifically for 3D datasets whose abstract representations are *spatially* and *semantically* nested. By spatially nested we mean that the abstractions are defined in the same spatial embedding but each uses a different amount of screen space such that more abstract representations are, generally, 'inside' less abstract ones. The nesting property allows us to use two-dimensional techniques to generate real-time halos which appear volumetric and visually separate the different visualizations. This allows us to create transitions between different abstractions which do not allow seamless geometrical transitions as demonstrated using different representations of fluid flow. We also incorporate lens-based navigation into the defined abstraction space allowing investigation of a different level of abstraction. Taken together, these techniques facilitate an intuitive continuous navigation of a set of nested abstractions of a given 3D visualization.

2. Related Work

Abstraction is a core principle in visualization and takes many forms, depending on the visualized data. Dedicated, controlled abstraction has been investigated not only in non-photorealistic rendering (e. g., [DS02, MDS09, WS94]) but also in visualization. In the field of information visualization many forms of intentional abstraction are used such as edge bundling (e. g., [Hol06]) or focus+context (e. g., [CM01]). In scientific and specifically illustrative visualization many "high-level visual abstractions" [RBGV08] are used.

Relevant for our own work are those high-level visual abstractions that not only show more or less relevant parts of a dataset in more or less detail but which can relate different visual representations (i. e., different abstraction levels) to each other. Duke [Duk04] describes this problem nicely and suggests linking different categories or representations to each other to uncover and understand the structure in a

dataset, naming molecular visualization as one example. Van der Zwan et al. [vdZLBI11] demonstrated such seamless transition between molecular abstraction levels in an interactive [vdZLBI11] and a spatially explicit [LVvdZ*11] manner. However, van der Zwan et al.'s [vdZLBI11] realization of abstraction level transitions requires that meaningful intermediate stages exist which is not the case for many forms of abstraction—a problem that we address in our own work.

We employ halos as one visual technique to visually separate layered elements and thus to enhance spatial perception. While halos enhance the spatial perception of the depicted objects in volume [BG07, IG98] and other visualization domains [EBRI09, TCM06], we employ them to support visual layering and thus related to their function of showing occlusion relationships in line-based techniques [ARS79, Elb95]. In addition, we use interactive lenses to locally explore our layered abstractions. Lenses are not only frequently used to support focus+context techniques [CM01] but also to interactively reveal otherwise hidden information, an approach pioneered by Bier et al. [BSP*93, BSP97]. Their Magic Lenses locally affect a 2D screen region using a user-selected operator. While lenses can be used in a 3D context to distort the projection [YCB05], they can also be used to specify non-view changes for a 3D scene in a separate 2D layer [HTE11, NIC07]. Our lenses have a similar function as we use them in a 2D layer over the 3D model to locally reveal relationships between abstraction layers, thus also relating to a number of smart visibility methods in visualization [VG05].

3. Visualization Model

We start with a dataset $d \in D$ and consider several visualizations of d, modeled as images $V_{1\leq i\leq N} : D \rightarrow \mathbf{R}^2$ and each producing a 2D image $A_i = V_i(d)$. We call these images *abstractions* of d if they represent the information in d on different levels of detail. We distinguish two abstraction types: *Semantic* abstractions A_i simplify the information in d by showing varying amounts of the information present in d using different visual representations. For example, a fluid flow volume $d \subset \mathbf{R}^3$ can be rendered as an entire flow volume using LIC [CL93, SH95], as stream LIC structures for a set of given streamlines [HA04], and as flow topology [TWHS03]; these are increasingly simplified semantic representations. *Sampling* abstractions reduce the amount of points produced by a given semantic abstraction A_i using data sampling. Rendering different numbers of streamlines, for example, are samplings of the streamline abstraction. We denote all S_i samplings of a semantic abstraction A_i by $A_i^j, 1 \leq j \leq S_i$ with $A_i^1 = A_i$ the most detailed and $A_i^{S_i}$ the coarsest sampling.

To be useful in an exploration scenario, abstractions must be described in terms of the amount of *simplification* they produce on some input dataset. In our model we assume that, for a given application domain (e. g., flow visualization), the abstraction set $A = \{A_i\}$ can be ordered in decreasing amount of provided simplification from the densest abstraction A_1 to the sparsest one A_N. We also require that simpler abstractions are visually *nested* within less simple ones, i. e. $A_j \subset A_i, \forall i < j$. This is often the case in scientific visualization where abstraction reduces the size and/or spatial dimensionality of the depicted visual elements while keeping them aligned in the space D. Our flow visualization scenario is such a case of nested abstractions: the topology is a part of the streamlines, these are nested within stream LIC representation, which in turn is a part of a LIC volume.

4. Navigating the Abstraction Space

Given an abstraction set A as just described, one typically wants to *navigate* A to get different types of insight which are best visible at different abstraction levels. One navigation option is to start with A_N (most abstract) and browse through A_i until A_1 (most detailed), optionally using spatial sampling to restrict the dense-data areas to zones of interest using, e. g., focus+context techniques. One can also start with the most detailed level A_1 and simplify the visualization to the coarsest level A_N is reached. During both navigation types, we call the level of the highest abstraction A_f being visualized the *focus* of the visualization: Given a user-selected f, we aim to produce a visualization combining all *context* abstractions $A_i, i < f$ and A_f in a single visualization such that all abstractions and their spatial nestings are shown. This will permit smooth navigation in the *combined* space of semantic and sampling abstractions (as introduced in Section 3).

Such navigations are typically realized by toggling the rendering of the elements A_i on and off. However, this creates sharp visual discontinuities in the transition, especially if the abstractions differ visually. Continuity can be added by smoothly interpolating the transparency or shape of consecutive A_i using fading or morphing while navigating through A. However, blending blurs the spatial nesting insight and can result in too high opacities when too many abstractions are blended. Morphing is not trivial for any pair of (nested) shapes, works only for shape pairs, and requires 3D shape representations rather than their 2D visualization results A_i.

We propose to create a continuous navigation function $Nav : A \times [0,1] \rightarrow \mathbf{R}^2$ to help navigation in the abstraction space. Given our set A of ordered, nested abstractions and a focus abstraction level $f \in [0,1]$, we combine all abstractions A to build a visualization V. As the user changes the focus level f, V continuously changes to show only A_1 (at $f = 0$), next show the focus abstraction A_f nested within lower abstractions as context, and finally A_N (at $f = 1$). The design of Nav should be such that it can be computed on any set of nested 2D or 3D abstractions, is continuous in f, clearly emphasizes the focus-context relation of nested abstractions, and is computed using only 2D image information instead of 3D shape information to achieve maximal performance.

We use an additive blending of the abstractions A_i in nesting levels (decreasing i) and compute the navigation function as $Nav(A, f) = \sum_{i=1}^{n} \alpha_i(f) A_i$ (see Fig. 1). The design of

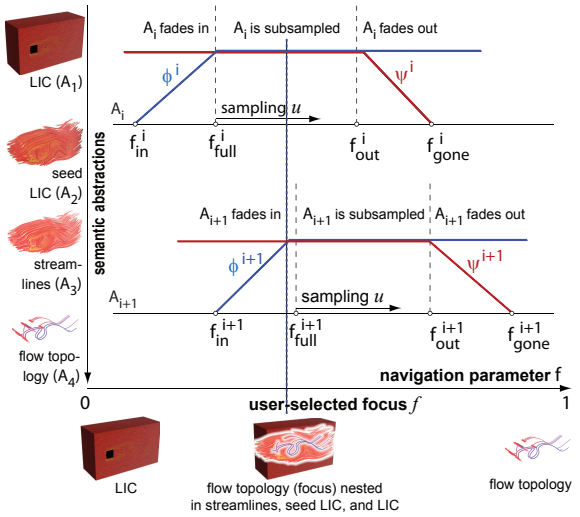

Figure 1: *Continuous navigation in a flow visualization abstraction space with four abstractions A_1–A_4.*

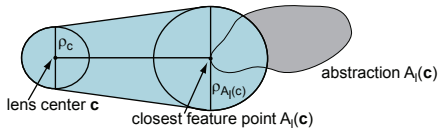

Figure 2: *Construction of the focus guided lens. Typical values are $\rho_{\mathbf{c}} = 5$ pixels and $\rho_{A_i(\mathbf{c})} = 90$ pixels.*

the blending factors $\alpha_i : [0,1] \to [0,1]$ is essential, we use $\alpha_i(f) = \phi_i(f) \cdot \psi_i(f) \cdot h_i(f)$. Here, $\phi_i : [0,1] \to [0,1]$ is the function used to fade in an abstraction A_i, $\psi_j : [0,1] \to [0,1]$ is the function responsible for fading out a (context) abstraction $A_j, j < i$, and $h_i : [0,1] \to [0,1]$ is the *halo function* used to create a halo around the selected abstractions. As fade-in function we use $\phi_i = \max\left(0, \min\left(1, \frac{f - f_{in}^{(i)}}{f_{full}^{(i)} - f_{in}^{(i)}}\right)\right)$ where $f_{in}^{(i)}$ is the focus value from which we start fading in abstraction A_i and $f_{full}^{(i)}$ is the value at which A_i is completely visible. In practice, we want to start fading in the next abstraction A_{i+1} when the current abstraction A_i is fully visible. Hence, we choose $f_{in}^{(i+1)} = f_{full}^{(i)}$. Similarly, we define the fade-out function $\psi_i = \min\left(1, \max\left(0, \frac{f_{gone}^{(i)} - f}{f_{gone}^{(i)} - f_{out}^{(i)}}\right)\right)$. Here, $f_{out}^{(i)}$ is the value for which we start fading out the context abstraction A_i and $f_{gone}^{(i)}$ is the value for which abstraction A_i is no longer visible. Similar to fading in, we start fading out abstraction A_{i+1} when the A_i is no longer visible and, thus, set $f_{gone}^{(i)} = f_{out}^{(i+1)}$. Also, we constrain $f_{out}^{i} > f_{full}^{i+1}$ such that A_i does not start fading out before A_{i+1} is fully in focus. Finally, we set $f_{full}^{1} = 0$, $f_{out}^{N} = 1$ so that we start with a fully focused A_1 and end with a fully focused A_N.

While combining ϕ_i and ψ_i allows us to fade abstractions in and out of view continuously, the resulting image will be unable to clearly show the *nesting* structure of the abstraction space: Depending on the specific abstraction image shapes and colors, it may be hard to see which result pixels belong to a (thin) abstraction being nested within a (larger) context abstraction, especially if both have similar colors. We therefore use the *halo function* h_i to generate halos around abstractions: $h_i(f) = \min\left(\left(DT_{A_{i+1}}/\delta\right)^{k_i(f)}, 1\right)$. In this function, $DT_\Omega : \mathbf{R}^2 \to \mathbf{R}^+$ is the distance transform of a 2D binary shape Ω, which gives, for any points $\mathbf{x} \in \mathbf{R}^2$, its distance to Ω [CC00]. DT is zero inside Ω and smoothly increases outside the shape. In our case, we construct such shapes by simply thresholding the rendered abstractions A_i into foreground (rendered) and background (not rendered) pixels. Having $DT_{A_{i+1}}$, we compute a halo around A_{i+1} by modulating the distance transform with a power function $k_i(f)$. The halo's width is limited to a maximal value of $\delta > 0$ pixels. The effect of the power function is to create a smoother transition from the context A_i of A_{i+1} than if linear distance functions were used. Finally, we set $k_i = \phi_i$ to increase the halo around the fading-in abstraction A_{i+1}, thus making it more prominent in its context A_i where it is nested. Perspective-like halos can easily be obtained by modulating the value of δ with the depth of A_{i+1} at each pixel.

The above process describes how *semantic* abstractions A_i are combined into a single image. However, as defined in Section 3, our input may contain sampled versions thereof. We integrate these smoothly in the above process by replacing, in the navigation function $Nav(A, f)$, each semantic abstraction A_i with its sampled version A_i^j, the sampling parameter j being controlled by the distance from the user-set focus f to the full-visibility f_{full}^i as $j = \frac{f - f_{full}^i}{f_{out}^i - f_{full}^i} S_i$. In other words, as the user increases f, the full-visibility abstraction is progressively simplified from A_i^1 to $A_i^{S_i}$ (coarsest variant). During this process, all remaining visualization elements stay the same (halo sizes, overall abstraction transparency). When f reaches f_{out}^i, the coarse abstraction $A_i^{S_i}$ is further faded out. For abstractions which have no level-of-detail representations, the process simply uses the unique representation A_i. This directly accommodates any number of semantic abstractions with any number of sampling representations thereof, effectively intertwining the navigation in the semantic and sampled spaces of abstractions.

5. Interactive Local Exploration

While this navigation facilitates an effective *global* abstraction space exploration, in many cases we are interested in getting local detail information. We thus also provide context-sensitive *local* lenses, whose goal it is to allow parts of an abstraction $A_{l>f}$ located inside the lens to become visible even when otherwise hidden due to the global abstraction level f. While a naïve implementation would change the blending factors $\alpha_i(f)$ close to the lens center, this would interfere with our transparent distance-based halos. In multi-layer

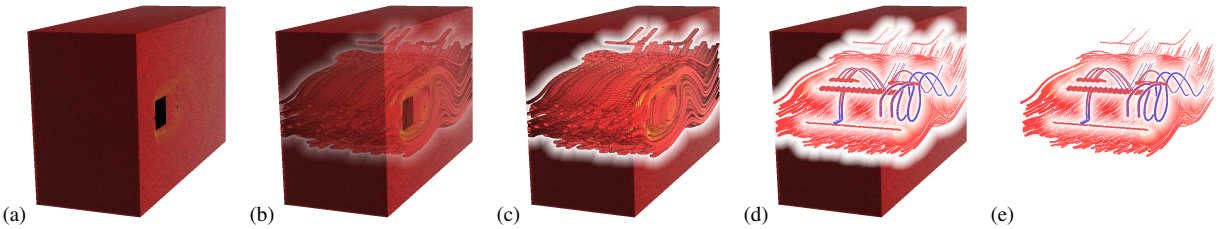

Figure 3: *Navigating the flow visualization abstraction space: (a)–(c) Introducing seed LIC as focus in the LIC volume, (d) has the flow topology as focus abstraction and the previous abstractions as context from which the LIC volume is removed in (e).*

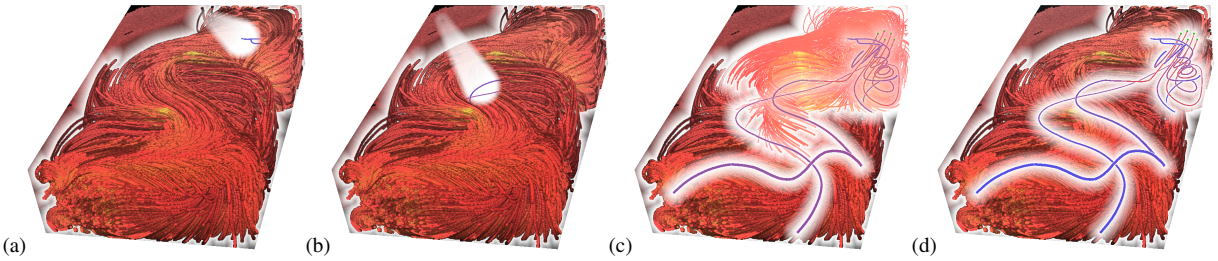

Figure 4: *Fluid flow abstracted (a)–(b) locally with a guided lens and (c)–(d) with sampling abstractions (streamline filtering).*

visualizations such as ours one also wants to see 'deeper' within the abstraction stack inside the lens *and* locate the parts of deeper-nested abstractions closest to to the lens.

For this, we use a *focus-guided lens*. Given a global abstraction level f, we combine revealing deeper-nested information at the lens center with revealing higher-abstraction structures $A_{l>f}$ close to it. We first locate the closest point $A_l(\mathbf{c})$ of abstraction A_l to the lens center $\mathbf{c}$. The point $\mathbf{c}$ can be directly computed as $A_l(\mathbf{c}) = FT_{A_l}(\mathbf{c})$. Here, $FT_{A_l} : \mathbf{R}^2 \rightarrow \mathbf{R}^2$ is the *feature transform* of the shape A_l [CC00]. The feature transform of a shape $\Omega \subset \mathbf{R}^2$ is defined as $FT_\Omega(\mathbf{x}) = \{\mathbf{y} \in \Omega | DT_\Omega(\mathbf{x}) = \|\mathbf{x}-\mathbf{y}\|\}$, i. e. the closest point $\mathbf{y} \in \Omega$ to a given target point $\mathbf{x}$, restricting ourselves to a one-point feature transform [EHP*11]. With the lens center $\mathbf{c}$ and closest abstraction point $A_l(\mathbf{c})$, we construct our focus-guided lens by multiplying the halo functions $h_{l>f}$ with the distance transform of a beam-like shape created by two circles connected by a trapezium (light blue in Fig. 2). As the lens is moved, it behaves similarly to a light beam that shows the shortest spatial path from the lens center to the desired A_l. This is useful as one does not need to fully remove (make transparent) *all* abstractions $A_{k<l}$ in order to discover A_l. Hence, one can stay at a desired semantic focus level f and use the lens to search for another desired $A_{l>f}$ in the vicinity of any point.

6. Implementation and Results

For the realization we only require a set of $N-1$ 2D images depicting the different context abstraction levels and one image depicting the abstraction in focus at the selected abstraction level, as our method works entirely in image space. These images are either generated on-the-fly or are precomputed if they do not change during the exploration. From these images we compute the soft halos for our nesting (within 10 ms on a modern graphics card) by employing a recent CUDA-based implementation [Tel11] of exact Euclidean distance transforms and feature transforms [CTMT10]. Finally, blending is implemented via OpenGL alpha blending. The entire process, including rendering LIC, seed LIC, streamlines, and precomputed topology, works at 5 frames per second on a MacBook with 2 GB RAM and an NVIDIA GeForce 320M graphics card with 256 MB RAM.

Fig. 3 shows our global abstraction applied to a direct incompressible Navier-Stokes flow simulation around a cylinder. Fig. 3(a)–(c) show three transition steps of stream LIC blended with the LIC volume separated by the halo around the new focus, *i.e.* a visualization with the streamline LIC as focus and the remainder (LIC) as context. A further abstraction (Fig. 3(d)) adds topology as the new focus. Finally, we remove the lowest abstraction (Fig. 3(e)). Our second example (Fig. 4) shows the local exploration of a different Navier-Stokes simulation, with Fig. 4(a) and (b) showing two lens locations. Fig. 4(c) and (d), finally demonstrate the inclusion of several sampling abstraction levels into the treatment.

7. Conclusion

In this paper, we have presented an abstraction technique for continuous navigation of nested abstraction levels. This navigation can be achieved on both global and local levels, using lenses for the latter. For both types of navigation, we use two-dimensional distance transforms to create smooth halos in the image domain which look like three-dimensional volumetric halos. As an example application, we used our technique to navigate different semantic and sampling abstractions of fluid flow visualization.

References

[ARS79] APPEL A., ROHLF F. J., STEIN A. J.: The Haloed Line Effect for Hidden Line Elimination. *ACM SIGGRAPH Computer Graphics 13*, 3 (Aug. 1979), 151–157. doi> 10.1145/800249.807437

[BG07] BRUCKNER S., GRÖLLER E.: Enhancing Depth-Perception with Flexible Volumetric Halos. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (Nov./Dec. 2007), 1344–1351. doi> 10.1109/TVCG.2007.70555

[BSP*93] BIER E. A., STONE M. C., PIER K., BUXTON W., DEROSE T. D.: Toolglass and Magic Lenses: The See-Through Interface. In *Proc. SIGGRAPH* (1993), ACM, New York, pp. 73–80. doi> 10.1145/166117.166126

[BSP97] BIER E., STONE M., PIER K.: Enhanced Illustration Using Magic Lens Filters. *IEEE Computer Graphics and Applications 17*, 6 (Nov./Dec. 1997), 62–70. doi> 10.1109/38.626971

[CC00] COSTA L., CESAR R.: *Shape Analysis and Classification: Theory and Practice*. CRC Press, 2000.

[CL93] CABRAL B., LEEDOM L. C.: Imaging Vector Fields using Line Integral Convolution. In *Proc. SIGGRAPH* (1993), ACM, New York, pp. 263–270. doi> 10.1145/166117.166151

[CM01] CARPENDALE M. S. T., MONTAGNESE C.: A Framework for Unifying Presentation Space. In *Proc. UIST* (2001), ACM, New York, pp. 61–70. doi> 10.1145/502348.502358

[CTMT10] CAO T.-T., TANG K., MOHAMED A., TAN T.-S.: Parallel Banding Algorithm to Compute Exact Distance Transform with the GPU. In *Proc. I3D* (2010), ACM, New York, pp. 134–141. doi> 10.1145/1730804.1730818

[DS02] DECARLO D., SANTELLA A.: Stylization and Abstraction of Photographs. *ACM Transactions on Graphics 21*, 3 (July 2002), 769–776. doi> 10.1145/566654.566650

[Duk04] DUKE D. J.: Linking Representation with Meaning. In *Posters of IEEE Visualization* (2004), IEEE Computer Society, Los Alamitos. doi> 10.1109/VISUAL.2004.66

[EBRI09] EVERTS M. H., BEKKER H., ROERDINK J. B. T. M., ISENBERG T.: Depth-Dependent Halos: Illustrative Rendering of Dense Line Data. *IEEE Transactions on Visualization and Computer Graphics 15*, 6 (Nov./Dec. 2009), 1299–1306. doi> 10.1109/TVCG.2009.138

[EHP*11] ERSOY O., HURTER C., PAULOVICH F., CANTAREIRA G., TELEA A.: Skeleton-Based Edge Bundling for Graph Visualization. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (Dec. 2011), 2364–2373. doi> 10.1109/TVCG.2011.233

[Elb95] ELBER G.: Line Illustrations $\in$ Computer Graphics. *The Visual Computer 11*, 6 (June 1995), 290–296. doi> 10.1007/s003710050022

[HA04] HELGELAND A., ANDREASSEN O.: Visualization of Vector Fields using Seed LIC and Volume Rendering. *IEEE Transactions on Visualization and Computer Graphics 10*, 6 (Nov./Dec. 2004), 673–682. doi> 10.1109/TVCG.2004.49

[Hol06] HOLTEN D.: Hierarchical Edge Bundles: Visualization of Adjacency Relations in Hierarchical Data. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (Sept./Oct. 2006), 741–748. doi> 10.1109/TVCG.2006.147

[HTE11] HURTER C., TELEA A., ERSOY O.: MoleView: An Attribute and Structure-Based Semantic Lens for Large Element-Based Plots. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (Dec. 2011), 2600–2609. doi> 10.1109/TVCG.2011.223

[IG98] INTERRANTE V., GROSCH C.: Visualizing 3D Flow. *IEEE Computer Graphics & Applications 18*, 4 (July 1998), 49–53. doi> 10.1109/38.689664

[LVvdZ*11] LUEKS W., VIOLA I., VAN DER ZWAN M., BEKKER H., ISENBERG T.: Spatially Continuous Change of Abstraction in Molecular Visualization. In *Abstracts of IEEE BioVis* (2011), IEEE Computer Society, Los Alamitos.

[MDS09] MI X., DECARLO D., STONE M.: Abstraction of 2D Shapes in Terms of Parts. In *Proc. NPAR* (2009), ACM, New York, pp. 15–24. doi> 10.1145/1572614.1572617

[NIC07] NEUMANN P., ISENBERG T., CARPENDALE S.: NPR Lenses: Interactive Tools for Non-Photorealistic Line Drawings. In *Proc. Smart Graphics* (2007), Springer-Verlag, Berlin, Heidelberg, pp. 10–22. doi> 10.1007/978-3-540-73214-3_2

[RBGV08] RAUTEK P., BRUCKNER S., GRÖLLER E., VIOLA I.: Illustrative Visualization: New Technology or Useless Tautology? *ACM SIGGRAPH Computer Graphics 42*, 3 (Aug. 2008), 4:1–4:8. doi> 10.1145/1408626.1408633

[SH95] STALLING D., HEGE H.-C.: Fast and Resolution Independent Line Integral Convolution. In *Proc. SIGGRAPH* (1995), ACM, New York, pp. 249–256. doi> 10.1145/218380.218448

[TCM06] TARINI M., CIGNONI P., MONTANI C.: Ambient Occlusion and Edge Cueing to Enhance Real Time Molecular Visualization. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (Sept./Oct. 2006), 1237–884. doi> 10.1109/TVCG.2006.115

[Tel11] TELEA A.: CUDA Skeletonization and Distance Transform Toolkit, 2011. http://www.cs.rug.nl/~alext/CUDASKEL.

[TWHS03] THEISEL H., WEINKAUF T., HEGE H.-C., SEIDEL H.-P.: Saddle Connectors – An Approach to Visualizing the Topological Skeleton of Complex 3D Vector Fields. In *Proc. IEEE Visualization* (2003), IEEE Computer Society, Los Alamitos, pp. 225–232. doi> 10.1109/VISUAL.2003.1250376

[vdZLBI11] VAN DER ZWAN M., LUEKS W., BEKKER H., ISENBERG T.: Illustrative Molecular Visualization with Continuous Abstraction. *Computer Graphics Forum 30*, 3 (May 2011), 683–690. doi> 10.1111/j.1467-8659.2011.01917.x

[VG05] VIOLA I., GRÖLLER E.: Smart Visibility in Visualization. In *Proc. CAe* (2005), Eurographics Association, Goslar, Germany, pp. 209–216. doi> 10.2312/COMPAESTH/COMPAESTH05/209-216

[WS94] WINKENBACH G., SALESIN D. H.: Computer-Generated Pen-and-Ink Illustration. In *Proc. SIGGRAPH* (1994), ACM, New York, pp. 91–100. doi> 10.1145/192161.192184

[YCB05] YANG Y., CHEN J., BEHESHTI M.: Nonlinear Perspective Projections and Magic Lenses: 3D View Deformation. *IEEE Computer Graphics & Applications 25*, 1 (Jan./Feb. 2005), 567–582. doi> 10.1109/MCG.2005.29